ИСП РАН

Верификация семантики линейной последовательности машинных инструкций

Алексей Вишняков

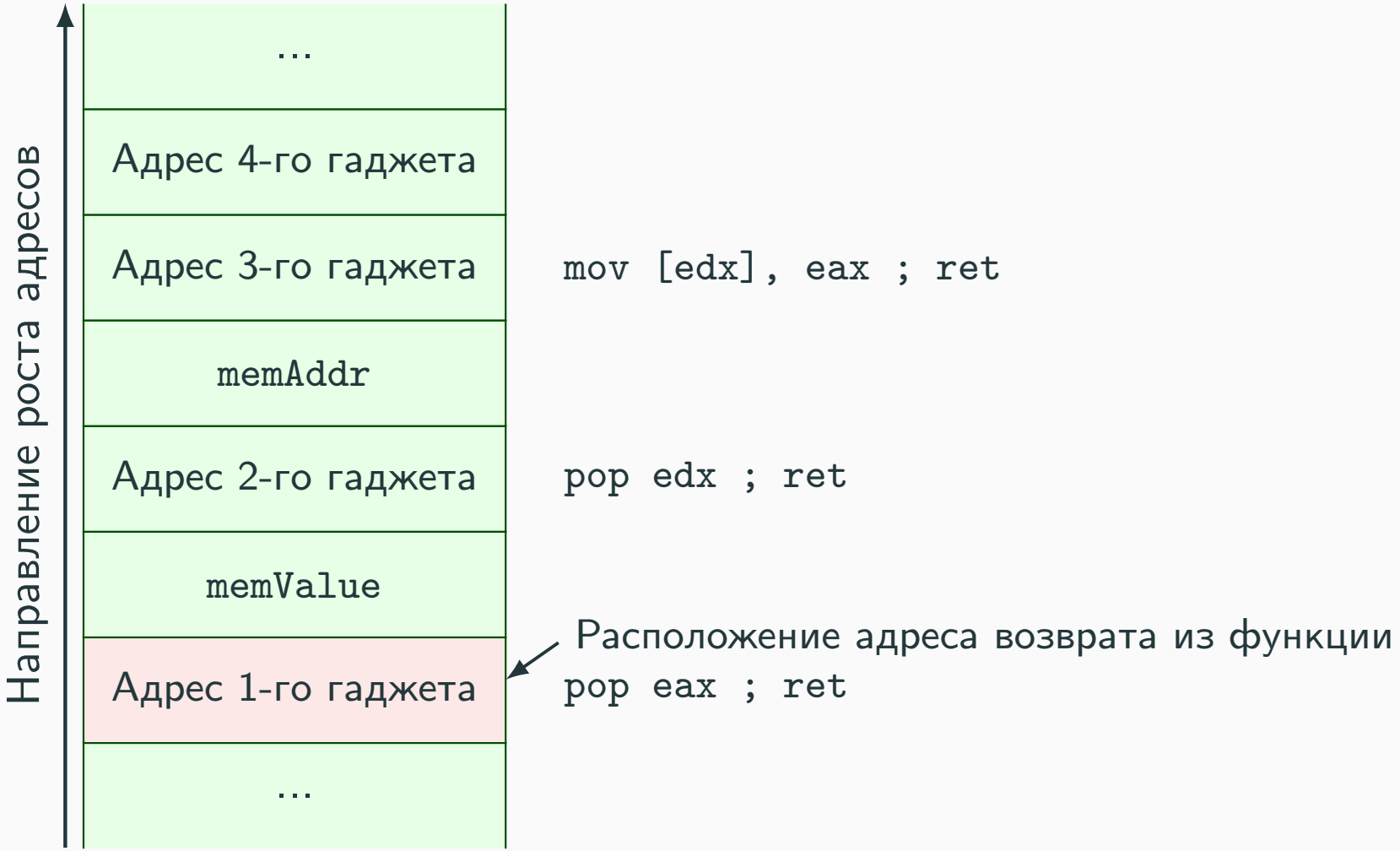
23 мая 2019

ИСП РАН

Возвратно-ориентированное программирование

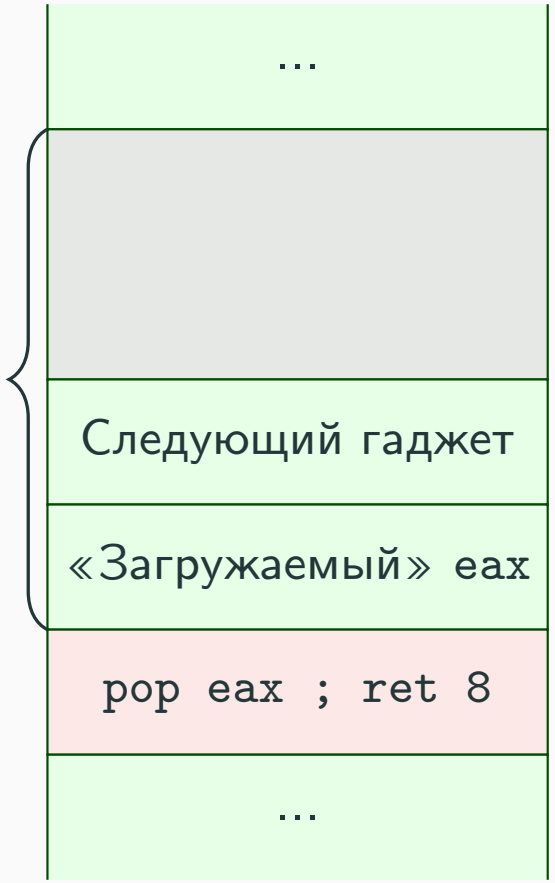
- **Возвратно-ориентированное программирование (ROP)** – атака повторного использования кода, позволяющая обходить DEP при наличии нерандомизированных областей памяти
- **Гаджет** – последовательность инструкций в нерандомизированной исполняемой области памяти, которая заканчивается инструкцией передачи управления
- Каждый гаджет выполняет некоторые вычисления (например, складывает значения двух регистров) и передает управление следующему гаджету
- Гаджеты связываются в цепочку, а их адреса размещаются от адреса возврата на стеке
- Таким образом, с помощью цепочки гаджетов можно выполнить некоторые вредоносные действия

Пример ROP цепочки
Запись значения `memValue` по адресу `memAddr`



Фрейм гаджета

- Введем понятие *фрейма гаджета* аналогично стековому кадру x86
- Размер фрейма
  `FrameSize = 16`
- Адрес следующего гаджета
  `NextAddr = [ESP + 4]`



Определение семантики гаджетов

- *Тип гаджета* описывается семантически с помощью постусловия, булева предиката, который должен быть всегда истинным после выполнения гаджета*
  - $MoveRegG : OutReg \leftarrow InReg$
  - $LoadConstG : OutReg \leftarrow M[SP + Offset]$
- Набор типов гаджетов задает новую архитектуру набора команд (ISA)
- Функциональность гаджета описывается набором параметризованных типов, которым принадлежит гаджет

```
push eax
pop  ebx
pop  ecx
ret
```

$MoveRegG : ebx \leftarrow eax$

$LoadConstG : ecx \leftarrow M[esp + 0]$

*Schwartz, Edward J., Thanassis Avgerinos, and David Brumley. "Q: Exploit Hardening Made Easy."USENIX Security Symposium. 2011.

Классификация гаджетов

- Классификация гаджета выявляет набор возможных типов и параметров, которым он соответствует
- Классификация производится на основе анализа эффектов выполнения гаджета на различных входных данных
- Составляется список типов и параметров с истинными постусловиями для всех выполнений гаджета с отличными входными данными

Результаты классификации гаджета

- Типы и параметры гаджета
- Список «испорченных» регистров (значения которых не сохраняются после выполнения гаджета)
- Информация о фрейме гаджета
  - Размер фрейма
  - Смещение ячейки с адресом следующего гаджета

```
0x08048061 : Asm : XCHG EBX, EAX ; XCHG EBX, EAX ; RET
0x08048061 : NoOpG
0x08048064 : Asm : MOV EAX, EBX ; RET
0x08048064 : MoveRegG : EAX <- EBX, AX <- BX, AH <- BH, AL <- BL
0x08048075 : Asm : POP EAX ; RET
0x08048075 : LoadConstG : EAX <- [ESP], AX <- [ESP], AH <- [ESP+1], AL <- [ESP] :
             NextAddr=[ESP+4], FrameSize=8
0x08048075 : ShiftStackG : ESP +<- 4
0x08048088 : Asm : ADD EAX, EBX ; RET
0x08048088 : ArithmeticG : EAX <- EAX + EBX, AX <- AX + BX, AL <- AL + BL
0x080480cb : Asm : MOV DWORD PTR [EAX - 00000456h], EBX ; RET
0x080480cb : StoreMemG : [EAX-0x456] <- EBX
0x080480da : Asm : ADD EAX, DWORD PTR [EBX] ; RET
0x080480da : ArithmeticLoadG : EAX +<- [EBX]
```

Цель работы

- Классификация гаджетов опирается на анализ эффектов выполнения гаджета на нескольких различных случайных входных данных
- Возможна неверная классификация гаджетов
- Целью работы является уточнение метода классификации гаджетов
- Необходимо формально доказать семантику гаджета для произвольных входных данных

Постановка задачи

Необходимо разработать и реализовать метод верификации семантики линейной последовательности машинных инструкций

Метод должен позволять по заданному постусловию верифицировать семантику ROP гаджета:

- Проверить, удовлетворяет ли гаджет определению семантики его типа с заданными значениями параметров
- Проверить список «испорченных» регистров
- Проверить размер фрейма и адрес следующего гаджета

Верификация гаджетов

- Классификация гаджета предоставляет набор постусловий, описывающих возможную семантику гаджета
- Верификация гаджета позволяет формально доказать истинность этих постусловий для произвольных входных данных
- Верификация гаджета реализована на движке динамической символьной интерпретации Triton, в который была добавлена поддержка символьных адресов

Метод верификации семантики гаджета

- Изначально всем регистрам присваиваются свободные символьные переменные
- Символьная память в начале представляет из себя пустой байтовый массив SMT Array
- Символьное состояние содержит отображение регистров в символьные переменные и текущее состояние символьной памяти
- Символьная интерпретация инструкции гаджета порождает SMT формулы над переменными и константами, а также обновляет символьное состояние в соответствии с операционной семантикой инструкции
- Работа с символьной памятью реализована через операции select и store (чтение и запись)
- Общезначимость формулы постусловия проверяется через невыполнимость ее отрицания

Пример верификации гаджета

ArithmeticLoadG : $rbx \leftarrow rbx + [rax]$

Шаг	**Символьное состояние**	**Инструкция**	**Множество формул**
initial	M, $rax = \phi_1$, $rbx = \phi_2$, $rcx = \phi_3$, $rsp = \phi_4$, $rip = \phi_5$	—	$S_0 = \emptyset$
1	$rcx = \phi_6$	`mov rcx, [rax]`	$S_1 = S_0 \cup \{\phi_6 = M[\phi_1]\}$
2	$rbx = \phi_7$	`add rbx, rcx`	$S_2 = S_1 \cup \{\phi_7 = \phi_2 + \phi_6\}$
final	$rip = \phi_8$, $rsp = \phi_9$	`ret`	$S_3 = S_2 \cup \{\phi_8 = M[\phi_4]$, $\phi_9 = \phi_4 + 8\}$
	Определение семантики		**Верификация**
verify	$(final(rbx) = initial(rbx) + initial(M[rax])) \wedge$ $(final(rip) = initial(M[rsp])) \wedge$ $(final(rsp) = initial(rsp) + 8)$		$\neg((\phi_7 = \phi_2 + M[\phi_1]) \wedge$ $(\phi_8 = M[\phi_4]) \wedge$ $(\phi_9 = \phi_4 + 8))$ is **UNSAT**

Результаты верификации гаджетов

- Удаляются неверно классифицированные гаджеты
- Верифицируются:
  - тип и параметры гаджета,
  - «испорченные» регистры,
  - информация о фрейме гаджета

```
ROPverifier - INFO - Remove id: 11341,
(0x45cb8a: neg eax ; sbb eax, eax ; and eax, ecx ;
           pop ebp ; ret),
type: move_reg, params: (in_reg: ECX, out_reg: EAX),
clobbered: (EAX, EBP, AX, AH, AL, BP),
next_addr: 4, frame_size: 8
```

Сравнение классификации и верификации гаджетов

- Верификатор дает уверенность в типе гаджета
- Разработка верификатора и результаты его работы позволили улучшить и отладить алгоритмы классификатора
  - Добавлены запуски классификации на граничных условиях 0 и -1
- Средняя скорость классификации ≈ 200 гаджетов в секунду, а верификации ≈ 2 гаджета в секунду

Количество гаджетов на каждой стадии: поиска, классификации, верификации на наборе файлов Windows 7 (64-разрядной) из 664 файлов

найдено	классифицировано	НЕ верифицировано
$14 \cdot 10^6$	$9 \cdot 10^6$	$65 \cdot 10^3$

Спасибо за внимание